

Atty. Docket No.: 11389-0036-999

**TITLE: SYSTEM AND METHOD ENABLING MULTIPLE
 PROCESSES TO EFFICIENTLY LOG EVENTS**

INVENTOR: Panagiotis Kougiouris and Mac Vu

APPENDIX A

**Pennie & Edmonds, LLP.
1155 Ave. of the Americas
New York, NY 10036
212-790-9090**

```

// vlogclient.idl : IDL source for vlogclient.dll
//

// This file will be processed by the MIDL tool to
// produce the type library (vlogclient.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";
//import "vlogserver.idl";

[
    object,
    uuid(EB012492-A4DB-11D1-BFDE-00201829472A),
    dual,
    helpstring("IHSLog Interface"),
    pointer_default(unique)
]
interface IHSLog : IDispatch
{
    // The level of event. The user
    // has a knob and uses a level to control how many events
    // to see
    typedef enum HSCLLogLevel {
        HSCLCritical    = 1, // use for more important events
        HSCLLevel1      = 1,
        HSCLError        = 2,
        HSCLLevel2       = 2,
        HSCLWarning      = 3,
        HSCLLevel3       = 3,
        HSCLInfo         = 4, // use the events that are less important
        HSCLLevel4       = 4
    } HSCLLogLevel;

    // The type of event. The user filters events based
    // on these switches. E.G. Log all the security
    // but not the operator events
    //
    // *** Look inside the library for definitions ***

    // The next two methods set the message catalog. Either of them could be used
    [id(2), helpstring("set the module")] HRESULT SetResourceFileName([in] BSTR lpFilena
me);

    [id(3), helpstring("set the module")] HRESULT SetResourceModule([in] long hModule);

    // The resource is assumed to have the following syntax:
    // logLevel, logMask, formatString
    [id(4), helpstring("log a message using resources")] HRESULT LogRes(
        [in] long nResourceId,
        [in, optional]VARIANT arg1,
        [in, optional]VARIANT arg2,
        [in, optional]VARIANT arg3,
        [in, optional]VARIANT arg4,
        [in, optional]VARIANT arg5);

    // [id(1), helpstring("log a message"), vararg] HRESULT Log([in] short logLevel, [in]
SAFEARRAY(VARIANT) psa);
    [id(1), helpstring("log a message using message catalogs")] HRESULT LogMC(
        [in] HSCLLogLevel logLevel,
        [in] LONG logMask, // HSCLLogType
        [in] long nMessageId,

```

```

[in, optional]VARIANT arg1,
[in, optional]VARIANT arg2,
[in, optional]VARIANT arg3,
[in, optional]VARIANT arg4,
[in, optional]VARIANT arg5);

```

```

[id(5), helpstring("log a message using a string")] HRESULT Log(
    [in] HSCLLogLevel logLevel,
    [in] LONG logMask, // HSCLLogType
    [in] BSTR bstrMessage,
    [in, optional]VARIANT arg1,
    [in, optional]VARIANT arg2,
    [in, optional]VARIANT arg3,
    [in, optional]VARIANT arg4,
    [in, optional]VARIANT arg5);
[id(6), helpstring("method ShowOptionsDialog")] HRESULT ShowOptionsDialog([in] LONG h

```

```

Wnd);
};

```

```

// This was added for languages like J++ that do
// not support optional arguments

```

```

{
    object,
    uuid(12DF1C10-8AFE-11d2-8E44-00104B79DD7C),
    dual,
    helpstring("IHSLog2 Interface"),
    pointer_default(unique)
}
interface IHSLog2 : IHSLog
{
    [id(7), helpstring("log a message using a string")] HRESULT Log0(
        [in] HSCLLogLevel logLevel,
        [in] LONG logMask, // HSCLLogType
        [in] BSTR bstrMessage);

    [id(8), helpstring("log a message using a string")] HRESULT Log1(
        [in] HSCLLogLevel logLevel,
        [in] LONG logMask, // HSCLLogType
        [in] BSTR bstrMessage,
        [in, optional]VARIANT arg1);

    [id(9), helpstring("log a message using a string")] HRESULT Log2(
        [in] HSCLLogLevel logLevel,
        [in] LONG logMask, // HSCLLogType
        [in] BSTR bstrMessage,
        [in, optional]VARIANT arg1,
        [in, optional]VARIANT arg2);

    [id(10), helpstring("log a message using a string")] HRESULT Log3(
        [in] HSCLLogLevel logLevel,
        [in] LONG logMask, // HSCLLogType
        [in] BSTR bstrMessage,
        [in, optional]VARIANT arg1,
        [in, optional]VARIANT arg2,
        [in, optional]VARIANT arg3);

    [id(11), helpstring("log a message using a string")] HRESULT Log4(
        [in] HSCLLogLevel logLevel,

```

```

        [in] LONG logMask, // HSCLLogType
        [in] BSTR bstrMessage,
        [in, optional] VARIANT arg1,
        [in, optional] VARIANT arg2,
        [in, optional] VARIANT arg3,
        [in, optional] VARIANT arg4);

};

[
    uuid(EB012485-A4DB-11D1-BFDE-00201829472A),
    version(1.0),
    helpstring("Healthcon Log Object (HSCLOG) 1.0")
]
library HSCLOG
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    typedef enum HSCLLogType {
        HSCLSecurity      = 1,
        HSCLOperator      = 2,
        HSCLPerformance   = 4,
        HSCLDebug         = 8,
        HSCLDebugDetail   = 16
    } HSCLLogType;

    [
        uuid(EB012494-A4DB-11D1-BFDE-00201829472A),
        helpstring("Logger Class")
    ]
    coclass Logger
    {
        [default] interface IHSLog;
        interface IHSLog2;
    };

    // The props UI object.

    [
        uuid(AC87A4FA-EA9B-11d1-8016-00201829472A),
        hidden,
        helpstring("CLogServer Prop UI Class")
    ]
    coclass PropertyPage
    {
        interface IUnknown;
    };
};

```